EXHIBIT 3

IN THE UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF VIRGINIA
ALEXANDRIA DIVISION

BMG RIGHTS MANAGEMENT (US) LLC, and ROUND HILL MUSIC LP, Plaintiffs, v. COX ENTERPRISES, INC., COX COMMUNICATIONS, INC., and COXCOM, LLC, Defendants.	Case No. 1:14-cv-1611 (LO/JFA)

SUPPLEMENTAL REBUTTAL REPORT OF CHRISTOPHER RUCINSKI

TABLE OF CONTENTS

Page

I. INTRODUCTION AND SHORT SUMMARY OF OPINIONS

1. My name is Christopher Rucinski. I am a computer scientist working at Elysium Digital, L.L.C. ("Elysium Digital"), a technical litigation consulting company located in Boston, MA. I have been retained by counsel for Cox Enterprises, Inc.; Cox Communications, Inc.; and CoxCom, LLC (collectively, "Cox") to supplement my Rebuttal Report of Christopher Rucinski ("Rucinski"), submitted on July 10, 2015. Pursuant to the Court's Order of July 17, 2015 (Dkt. No. 156), I submit this supplemental report in response to (1) source code that Rightscorp, Inc. ("Rightscorp") produced after July 10, 2015; (2) torrent payload samples that Rightscorp produced at the July 3, 2015 deposition of Greg Boswell on two hard drives; (3) the Reply Report of Barbara Frederiksen-Cross ("Frederiksen-Cross"), submitted on July 24, 2015, and corrected on July 27, 2015; and (4) the Court-ordered Rule 30(b)(6) depositions of Greg Boswell and Robert Steele on July 29, 2015.

2. My opinions are described in detail in the subsequent sections of this report, and my opinions can be summarized as follows:

i. Rightscorp's failure to implement a version control system is unreasonable and inconsistent with industry standards for software development.

ii. Rightscorp's source code and data production remains inadequate.

iii. The apparent operation of past versions of Rightscorp's source code introduces further concerns about the reliability of Rightscorp's software for detecting instances of alleged infringement during the relevant time period.

iv. The torrent payload samples produced by Rightscorp during Greg Boswell's deposition on July 3, 2015, are not a reliable evidentiary record of music that BitTorrent peers allegedly offered.

v. Frederiksen-Cross continues to materially misstate and mischaracterize the operation of Rightscorp's source code.

3. This report summarizes my current opinions, which are subject to change depending on ongoing discovery and additional information. A list of the materials I considered in preparing this report is attached as Exhibit B.

II. QUALIFICATIONS AND COMPENSATION

4. I graduated *cum laude* with an A.B. in Computer Science from Princeton University in 2010, and in 2015 I obtained GCFE (GIAC[1] Certified Forensic Examiner) certification. In 2010 I began working at Elysium Digital, and in that capacity I have worked on more than 50 technical matters to date. Many of those technical matters have involved analysis of Java source code, and I have more than ten years of experience writing and analyzing Java source code. I am also

[1] "GIAC" stands for Global Information Assurance Certification: http://www.giac.org

well-versed in a number of other programming languages including but not limited to Python, C, Objective-C, Perl, and SQL.

5. I have provided consulting services for the Federal Trade Commission, and I provided deposition testimony in 2013 in the matter of *Shurtape Technologies, LLC et al. v. 3M Company*, U.S. District Court of Western North Carolina (Case No. 5:11-cv-00017) (trademark dispute related to consumer statements on the Internet).

6. My CV is attached as Exhibit A.

7. Elysium Digital is being compensated for my time at my standard rate of $340 per hour. Elysium Digital is also being compensated at varying rates ranging from $120 to $520 per hour for the work of additional Elysium Digital employees who are working at my direction. All compensation described above does not depend on the outcome of this case.

III. RIGHTSCORP'S FAILURE TO IMPLEMENT A VERSION CONTROL SYSTEM IS UNREASONABLE AND INCONSISTENT WITH INDUSTRY STANDARDS FOR SOFTWARE DEVELOPMENT.

8. Greg Boswell has testified that no version control system is or has ever been utilized by Rightscorp. In his July 29, 2015, deposition, Boswell attributed Rightscorp's failure to adopt a revision control system to Rightscorp's adherence to the Extreme Programming ("XP") software development

methodology.[2] The XP software methodology is one type of "agile" software development. The XP software development methodology does not advocate against utilizing a version control system, nor does any other type of agile software development methodology. To the contrary, as a general principle, the XP software development methodology (and agile software development methodologies more broadly) promotes the use of a source code version control system by software developers as a best practice, even in the context where only a single developer is developing software for the project. For example, Extreme Programming with Ant: Building and Deploying Java Applications with JSP, EJB, XSLT, XDoclet, and JUnit by Glenn Niemeyer and Jeremy Poteet (2003) states:[3]

> No matter what the size of the development team is, version control plays an important role in the process. At one level, you can think of version control as a part of an automated backup process. Beyond that basic level, version control enables development team members to work in conjunction with each other, regress to earlier versions, and treat a group of files as a single unit. We believe that it's important at the outset to set up a version-control practice. Version control is one of those areas that isn't as interesting as architecture or design, and it can easily be neglected. This is exactly why we think it's important to set up such a system initially; that way, it isn't postponed or neglected entirely.

[2] Deposition of Greg Boswell, July 29, 2015, 339:22 – 25

[3] http://www.informit.com/articles/article.aspx?p=32052&seqNum=7; page 84 in the physical book.

9. As discussed in Rucinski ¶39, the use of version control is an industry standard. The reasonable decision for any company whose product or service is based primarily in software is to adopt some method of version control; it is a remarkably inexpensive practice to adopt that conveys many benefits. For example, in "What is Version Control? Why is it Important for Due Diligence," Stuart Yeates explains:[4]

> Developers may wish to compare today's version of some software with yesterday's version or last year's version. Since version control systems keep track of every version of the software, this becomes a straightforward task. Knowing the what, who, and when of changes will help with comparing the performance of particular versions, working out when bugs were introduced (or fixed), and so on.

The decision to use some method of version control is made even easier by the myriad free open source version control systems, including Git, Concurrent Version System ("CVS"), and Subversion ("SVN"), all three of which Boswell has testified to using at some point in his career.[5]

10. Frederiksen-Cross at ¶21 disagrees with the statements in Rucinski at ¶39 regarding Rightscorp's lack of a code version control system being unreasonable. Frederiksen-Cross at ¶21 states:

> This is especially true for a company that is developing source code for its own use as opposed to software that will be

[4] http://oss-watch.ac.uk/resources/versioncontrol

[5] Deposition of Greg Boswell, July 29, 2015, 335:16 – 338:1

> deployed to a wide customer base that may be using different releases of their product. Rightscorp also has only a small number of individuals involved in its source code development, and these individuals work closely together with frequent opportunities to discuss proposed code changes.

Frederiksen-Cross does not mention the broader issue, which is that while Rightcorp may have a small software development team that does not distribute its software to customers, it is and had been developing software for application in a legal context – and in particular, for evidence gathering purposes – where the historical operation of that source code would be relevant. The lack of a coherent historical code production that spans the entirety of the relevant time period makes it impossible to know how the source code functioned for vast spans of time. This is further exacerbated by the fact that the XP software development methodology which advocates for frequent release cycles. See, e.g. "Some teams deploy new software into production every day. At the very least you will want to get new software into production every week or two."[6]; "Second, XP teams release to their end users frequently as well. *XP Web projects release as often as daily*, in house projects monthly or more frequently"[7] (emphasis added). Since the source code changes so frequently, as often as daily, even a complete record of all the source code on a given day might only accurately reflect the operation of the whole

[6] http://www.extremeprogramming.org/rules/releaseoften.html

[7] http://ronjeffries.com/xprog/what-is-extreme-programming/

software system on that day. As described later in this report, Rightscorp has not even managed to produce a complete historical version for any point in time, and many of the historical source code files it did produce were never in production. In general, but also specifically for a company like Rightscorp, it is unreasonable to neglect the establishment of a version control system.

IV. OPERATION OF NEWLY PRODUCED RIGHTSCORP SOURCE CODE

11. On July 15, 2015, and July 16, 2015, I received additional source code production from Rightscorp. The operation of the source code in that production is described in this section.

A. New Production that Appears to Relate to the Most Current Version of Source Code that Rightscorp has Produced

1. SQL Stored Procedures: "tracker" and "ReEvaluateFullFilesBeing0"

12. Rightscorp has produced a number of stored procedures[8] from their database over the course of this litigation. Stored procedures are not stored in separate files but are instead stored in the database itself. Boswell has testified that, in collecting these stored procedures for this litigation, he copied their contents into text files and then produced the text files.[9] Because these produced text files were created and could have been modified independently from the actual stored

[8] See, e.g. https://en.wikipedia.org/wiki/Stored_procedure

[9] Deposition of Greg Boswell, July 29, 2015, 442:5 – 443:5

procedures in the database, the filesystem level metadata about the produced text files, such as creation time and last modification time, do not reflect the creation and last modification time of the corresponding stored procedures. I do not know the exact configuration of Rightscorp's database. That said, as a general matter, it is possible to retrieve the creation and modification times for stored procedures in a MySQL database by using the "SHOW PROCEDURE STATUS" database command,[10] and it is likely that Rightscorp could have used that command or a similar command corresponding to their database configuration to accurately produce metadata for stored procedures.

13. Stored procedures can be executed manually, and they can also be executed by database events.[11] Database events are procedures that run on a set schedule; they can be programmed to run stored procedures on a recurring schedule, e.g. every 15 minutes or every day. Database events can also specify a date and time before which they will not execute (I will refer to such dates as "start dates"); for example, one might use this technique to specify a database event that will execute a stored procedure once every 15 minutes but will not begin until next

[10] See https://dev.mysql.com/doc/refman/5.0/en/show-procedure-status.html

[11] See, e.g. http://www.sitepoint.com/database-triggers-events/

Monday. Stored procedures can be executed manually even if there are one or more database events that cause them to be executed automatically.[12]

14. For instance, as part of the source code produced by Rightscorp on July 15, 2015, Rightscorp included the definition of a database event that runs a stored procedure called "tracker" once per hour.[13] (As described in Rucinski ¶36, the "tracker" stored procedure populates the *GetThisInfringer* database table that is used by the `SampleIt2.java` program to select a .torrent file to download the corresponding payload from a given peer.) The start date for the database event is September 25, 2014, which indicates that the "tracker" stored procedure began executing on an automated schedule on than that date.

15. Even though the database event indicates that the "tracker" stored procedure began running automatically on September 25, 2014, it could have been executed manually at any time, both before or after this date. In fact, Boswell testified that this stored procedure was run manually by a Rightscorp employee "at a[] given time during the day" as early as February 2014.[14]

[12] Deposition of Greg Boswell, July 29, 2015, 444:19 – 445:8

[13] Rightscorp 07-15-15/StoredProcedures/StoredProcedures/evennts_runINSIDEoftheDatabaseEngine/botfeeder2

[14] Deposition of Greg Boswell, July 3, 2015, 197:8 – 198:2, 252:14 – 19

16. As described in Rucinski ¶43, `FullFileFix.txt` contains a database event that calls a stored procedure named "ReEvaluateFullFilesBeing0" every fifteen minutes, and for the purposes of my analysis I have assumed that that the stored procedure named "ReEvaluateFullFilesBeing0Hold" defined in `FullFilesBeing.txt` is therefore called every fifteen minutes despite the discrepancy in the name of the stored procedure. Boswell has testified that he occasionally changes the name of this stored procedure to cause it to not run in an automated fashion,[15] which appears to be the cause of this source code production discrepancy.

17. As described in Rucinski ¶43, the stored procedure shown in `FullFilesBeing.txt` defines the threshold of 10% for the number of bits in a bitfield that are required to be advertised by a BitTorrent peer at an IP address before Rightscorp's software will consider the peer to be offering an allegedly infringing file. Boswell also testified that the threshold for the number of bits in a bitfield that are required to be advertised by a BitTorrent peer at an IP address before Rightscorp's software will consider the peer to be offering an allegedly infringing file was reduced from 100% to 10% in December 2014.[16]

[15] Deposition of Grep Boswell, July 29, 2015, 445:21 – 446:18

[16] Deposition of Greg Boswell, July 3, 2015, 60:11 – 18

18. The start date of the database event shown in `FullFileFix.txt` is December 2, 2014. That said, just as the "tracker" stored procedure was executed manually, potentially as frequently as daily,[17] before it was executed in an automated fashion, so too could the stored procedure shown in `FullFilesBeing.txt` have been executed manually and perhaps just as frequently before it was executed in an automated fashion.[18] The start date for the database event that periodically executes the stored procedure shown in `FullFilesBeing.txt` merely indicates when the execution of that stored procedure could have been automated.[19] It is also possible that the start date of the database event used to be set to an earlier date in an earlier version of the database event, indicating an earlier date of initial automation; because there is no record of historic versions of the database event, it is impossible to determine this one way or another.

[17] Deposition of Greg Boswell, July 3, 2015, 197:8 – 198:2

[18] Depositon of Greg Boswell, July 29, 2015, 444:25 – 445:20

[19] Rightscorp 07-09-15/FullFileFix.txt, RIGHT_SC00000403

19. As discussed more generally above, last modification dates[20] of the files `FullFilesBeing.txt` (June 15, 2015) and `FullFileFix.txt` (July 9, 21015) provide no useful information regarding when they might have been first used in the Rightscorp system. Therefore, I have seen no evidence in the Rightscorp source code production that conclusively demonstrates whether or not the "ReEvaluateFullFilesBeing0Hold" stored procedure, as shown in `FullFilesBeing.txt`, was executed manually or not executed manually by a Rightscorp employee at any time. I also therefore explicitly disagree with Frederiksen-Cross at ¶12 in footnote #1 that any creation/modification dates associated with the `FullFileFix.txt` file could be used to accurately determine the date at which the stored procedure shown in that file was first used; Frederiksen-Cross likely intended to instead rely on the start date of the database event in this analysis.

2. Source Code Relating to Accounting

20. As part of the source code produced by Rightscorp on July 15, 2015, Rightscorp produced for the first time in this litigation several files related to the

[20] The last modification dates of files I discuss here and elsewhere in this report refer to "Modify" date and time reported by the NTFS filesystem of the Source Code Review computer on which I reviewed the Rightscorp source code. See, e.g., http://forensicswiki.org/wiki/New_Technology_File_System_(NTFS)#Time_Stamps

payment by ISP subscribers of settlement offers sent out by Rightscorp. These files include web pages that allow a user to view and update information in the *accounting* database table, a web page that displays the daily average, month-to-date, and year-to-date amounts of money collected from ISP subscribers, and the web page shown to an ISP subscriber if they click the link in the settlement offer.

21. The web page shown to an ISP subscriber if they click the link in a settlement offer is generated by the source code in `Payment.java`[21] The link in a settlement offer is "https://secure.digitalrightscorp.com/settle/<infractionguid>", where "<infractionguid>" is an identifier for the alleged infringement.[22] The web page contains a form that the subscriber fills out with his or her personal information (e.g. name and address) and credit card information. When the user submits the form, the `Payment.java` program authorizes the credit card through the Authorize.Net payment processor[23] and then updates the *infractions* database table with the subscriber's personal information and credit card information (see line 198).

[21] Rightscorp 07-15-15/ACCOUNTING1/ACCOUNTING/Payment.java

[22] See 2015.05.26 – Additional Rightscorp Source Code/CoxEmailCode/CEmail.java at line 75, RIGHT_SC00000018; Rightscorp 07-15-15/ACCOUNTING1/ACCOUNTING/servletmapings, RIGHT_SC00000539

[23] See Rightscorp 07-15-15/ACCOUNTING1/ACCOUNTING/authorize.java, RIGHT_SC00000524

22. A second feature of the `Payment.java` program is that it records information about any visitor to the web page that it generates. Such a visitor could be the recipient of a settlement offer who has clicked the link in the settlement offer email, but it could also be any other user who happens to access the URL. The `Payment.java` program inserts into the *visitor* database table the "infractionguid," a unique identifier for the alleged infringement (taken from the URL of the web page), the visitor's IP address, and other information about the visitor, such as information about the web browser that the visitor is using (see line 288).

23. In order to see a list of all the alleged infringements associated with a particular IP address, a Rightscorp employee can use the `PartialPays.jsp` web page.[24] The `PartialPays.jsp` web page selects all rows from the *PartialPays* database table. The *PartialPays* database table is periodically recreated and populated with data from the *infractions* database table that includes the IP address, name, email address, and phone number of someone who has paid a settlement fee to Rightscorp on one or more occasions and has other alleged infringements for which they have not paid.[25] The `PartialPays.jsp` web page lists each such person along with a button that takes the Rightscorp employee to the

[24] Deposition of Robert Steele, July 29, 2015, 13:10 – 14:11; Deposition of Greg Boswell, July 29, 2015, 460:8 – 460:20, 462:2 – 463:21

[25] Rightscorp20150627/2015-627/admin/sqltext, RIGHT_SC00000380

`FindIt.jsp` web page.[26] Based in part on testimony from Robert Steele,[27] my understanding is that the purpose of the `PartialPays.jsp` web page is to allow Rightscorp employees to contact individuals, who have paid to settle some alleged copyright infractions, in order to solicit payment for other alleged copyright infractions that are associated with the same IP address and port that a given individual already settled with a payment.

24. The `FindIt.jsp` web page lists all entries from the *infractions* database table for a given condition.[28] If a Rightscorp employee comes to the `FindIt.jsp` web page by clicking one of the buttons on the `PartialPays.jsp` web page, then the `FindIt.jsp` web page will list all the entries from the *infractions* database table matching the given IP address and port number (forms at the top of the FindIt.jsp web page also allow a Rightscorp employee to list all entries from the *infractions* database table for a given infractionguid[29]). No other information besides IP address and port number is used to associate unpaid alleged infringements with previously paid alleged

[26] Rightscorp20150627/2015-627/admin/PartialPays.jsp, RIGHT_SC00000367-368

[27] Deposition of Robert Steele, July 29, 2015, 13:10 – 14:11

[28] Rightscorp20150627/2015-627/admin/FindIt.jsp, RIGHT_SC00000350-361

[29] Deposition of Robert Steele, July 29, 2015, 12:18 – 12:22

infringements[30] even though IP addresses can be reassigned to different individuals over time.

3. Newly Produced Source Code Relating to Current Version Notice Generation Process

25. As part of the source code produced by Rightcorp on July 15, 2015, Rightscorp produced a file named `CPost.html`.[31] This HTML file contains a single form[32] that allows a user to select a number of notifications of claimed infringement to be generated at a time.[33] The handler for this form is the previously produced `CEmail.java`. When the user submits the form on `CPost.html`, `CEmail.java` generates groups of emails from its selection from a combination of the *infractions*, *copyrights*, and *EmailRegestry* database tables; each group of emails comprises no more emails than the number selected by the user using `CPost.html`.[34] Boswell testified that the purpose of being able to set the size of each group of emails is to be able to balance the load on the computer that is

[30] Deposition of Greg Boswell, July 29, 2015, 475:19 – 476:19

[31] To the extent that Frederiksen-Cross at ¶27 suggests that `CPost.html` should have been analyzed in Rucinski, I will explicitly note that `CPost.html` was not produced until after Rucinski was submitted.

[32] http://www.w3schools.com/html/html_forms.asp

[33] The values that may be selected range from 1 to 50,000.

[34] Deposition of Greg Boswell, July 29, 2015, 360:19 – 25

sending the emails.[35] Rucinski ¶31 also describes the operation of the `CEmail.java` program. Boswell has testified that the `CEmail.java` program is started only by using the `CPost.html` file, which is used manually.[36] Boswell has testified that Cox is the only ISP for which such a manual process exists for sending notifications of claimed infringement, and that this difference was a business decision.[37] I have not seen any source code related to sending notifications of claimed infringement for ISPs other than Cox, and so I am unable to comment on the differences between these two processes.

26. Frederiksen-Cross at ¶17[38] discusses the mechanism by which notifications of claimed infringement are generated. In particular, this paragraph states:

> The limitation for one infringement notice per day per file per IP address/port number is provided through the use of the *[E]mail[R]egestry* table. This table is defined by SQL statements in [the] file Rightscorp201506091\20150609\sqs20150609. The definition for this [] table includes a "UNIQUE KEY" that is comprised of the IP address, Port, date, and filename. The purpose of a UNIQUE KEY is to ensure that no two records within the database table have the same value for the key.

[35] Deposition of Greg Boswell, July 29, 2015, 361:16 – 23

[36] Deposition of Greg Boswell, July 29, 2015, 361:1 – 6

[37] Deposition of Greg Boswell, July 29, 2015, 371:10 – 372:4

[38] The "UNIQUE KEY" restriction of the *EmailRegestry* database table is also referenced in Frederiksen-Cross at ¶¶8, 16, and 26.

It was an error in Rucinski to have overlooked the significance of line 131 of Rightscorp201506091\20150609\sqs20150609, the one line that defines the "UNIQUE KEY" as described in Frederiksen-Cross at ¶17. The operation of this line is inconsistent with the five notifications of claimed infringement that I discuss in ¶45 of Rucinski, where each of the five notifications of claimed infringement concerns the same IP address, port number, filename, and date (February 16, 2012). It is possible that the restriction on the records of the *EmailRegestry* database table were not in place at the time these five notifications of claimed infringement were generated, and that restriction was added only at some indeterminate date thereafter. Robert Steele has testified that such a change occurred in "late 2011, early 2012,"[39] but since Rightscorp has not implemented any version control system, it is not possible to know exactly when such a change was implemented. Further, Boswell has no recollection of when this change was implemented.[40]

27. However, there is a subtle distinction in this functionality. As discussed above, email notifications of claimed infringement are sent out by the `CEmail.java` program, which draws information from the *EmailRegestry* database table and is executed only manually. Boswell testified that sometimes the

[39] Deposition of Robert Steele, July 29, 2015, 14:22 – 15:2

[40] Deposition of Greg Boswell, July 29, 2015, 449:22 – 25

`EmailRegestry` table can accumulate multiple incidents of claimed infringement for the same IP, port, and filename. When the `CEmail.java` program is then executed, multiple notifications of claimed infringement for that IP, port, and filename will then be sent out very quickly, quickly enough to be sent out on the same day.[41,42] Therefore, to the extent Frederiksen-Cross states that ISPs are limited to receive at most one email per day related to a given IP, port, and filename, I disagree.[43] The operation of the Rightscorp system in this regard also does not agree with Robert Steele's stated conception of how the system works; Robert Steele testified that the relevant business rule limited sending notifications of claimed infringement to "one notice per day per torrent per copyright."[44]

28. Furthermore, since the restriction in the *EmailRegestry* database table is limited to only IP, port, day, and filename, it is possible that multiple identical songs with the same IP and port could still be successfully inserted into the *EmailRegestry* database table. This could happen if two such identical files for a song existed with different filenames, which might happen, e.g., if one file was from a "Greatest Hits" compilation and the other was from a regular album.

[41] Deposition of Greg Boswell, July 29, 2015, 453:25 – 455:25

[42] See also Exhibits 20 – 24 from the Deposition of Greg Boswell, July 29, 2015: RGHTS19335897, RGHTS19324868, RGHTS19281664, RGHTS19344597, RGHTS19359397

[43] See Frederiksen-Cross at ¶¶8, 16, 17, and 26

[44] Deposition of Robert Steele, July 29, 2015, 9:15 – 24

Furthermore, the restriction on the *EmailRegestry* database table does not limit entries pertaining to the same allegedly infringed copyrighted work from a given IP and port on a given day. My understanding is that the copyrighted works at issue in this litigation are musical compositions, not sound recordings, and that multiple different recordings can pertain to the same musical composition. So, for instance, a regular version of a song and an acoustic version of a song would both embody the same copyrighted musical composition. Assuming two such songs had different filenames, they could both be added to the *EmailRegestry* database table with the same IP, port, and day. Therefore, to the extent Frederiksen-Cross states that ISPs are limited to receive at most one email per day per *song* or *copyrighted work* per IP address, and port, I disagree.[45] Rather, the restriction on the *EmailRegestry* database table does not permit duplicate entries corresponding to the same IP, port, day, and *filename*.

B. Substantive Partial Productions of Past Versions of Rightscorp Source Code

29. Rightscorp has produced what appears to be two substantive partial productions of past versions of source code. Each substantive partial production consists of a subset of the files that were required to operate the full Rightscorp system at a previous time. The first substantive partial production comprises files

[45] See Frederiksen-Cross at ¶¶8, 16, 17, and 26

with last modification dates ranging from September 3, 2011, to April 1, 2013, and in this report I will refer to the source code in this substantive partial production collectively as "RC-v1."[46] The second substantive partial production comprises files that all have a last modification date of September 9, 2013, and in this report I will refer to the source code in this substantive partial production collectively as "RC-v2."[47] Barbara Frederiksen-Cross was provided the source code corresponding to RC-v1 on June 20, 2013 and the source code corresponding to RC-v2 on November 18, 2013 during her technical review of Rightscorp's system; both of these dates are consistent with the most recent last modification times in RC-v1 and RC-v2.[48] In this report I will refer to RC-v1 and RC-v2 collectively as "RC-Historical," and I will refer to code that was produced by Rightscorp before July 15, 2015, or that otherwise appears to relate to the most recent version of Rightscorp source code that has been produced as "RC-Current."[49] For the purposes of this report I assume that the last modification dates of the files in RC-Historical accurately reflect the last time each of those files was modified, and based on that

[46] RC-v1 comprises the files in the following source code directory: Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent

[47] RC-v2 comprises the files in the following source code directory: Rightscorp 07-15-15/RC20130909/jBittorentAPI-1/com/boswell/torrent

[48] July 24, 2015 Letter from Stephanie Roberts to Brian Buckley

[49] RC-Current comprises at least the files in the following source code directory: Rightscorp Source Code/com/boswell/torrent

assumption, all the source code files in RC-v1 predate all the source code files in RC-v2. The last modification date for a given source code file is particularly relevant because assuming that the given file is being used in production, the last modification date will indicate the earliest date that that file was used in production with its current source code defining its operation.

30. A directory of additional files necessary for the operation of RC-v1 and RC-v2 has also been produced. This directory comprises the following source code files: `Detected.jsp`, `DownloadRequest.jsp`, `InfringementFinder.sh`, `IPRequest.jsp`, `TFT.jsp`, and `TorrentRequest.jsp`; in this report I will refer to these source code files collectively as "RC-JLI."[50] The last modification dates for the files in RC-JLI range from May 29, 2012 to July 2, 2013. Barbara Frederiksen-Cross was provided with the source code corresponding to RC-JLI on July 15, 2013 during her technical review of Rightscorp's system; this is consistent with the most recent last modification times in RC-JLI.[51] I cannot determine whether RC-JLI was used with RC-v1, RC-v2, both, or neither at least because the last modification times of the files in each production are so varied. For the purpose of this report I have analyzed RC-v1 and RC-v2 under the assumption that both utilize the files in RC-JLI since

[50] RC-JLI comprises the files in the following source code directory: Rightscorp 07-15-15/jli transmittal 7-2-131/jli transmittal 7-2-13/

[51] July 24, 2015 Letter from Stephanie Roberts to Brian Buckley

the set of files in RC-JLI do not appear in either RC-v1 or RC-v2. With that assumption, RC-v1 and RC-v2 operate in substantially the same way, though I have noted differences in their operation below. RC-v1 and RC-v2 also operate in substantially the same way as RC-Current, though I have noted below ways in which their operation differs.

1. Ingestion of Copyrighted Works and Selection of .torrent Files to Monitor

31. Rucinski ¶¶14 – 18 discusses the operation of RC-Current with respect to (1) ingestion of copyrighted works, (2) finding .torrent files on the Internet, and (3) filtering downloaded .torrent files based on matching the artist and title of copyrighted works against the filename in order to subsequently monitor the filtered .torrent files. I have seen no source code in RC-v1 or RC-v2 that suggests RC-v1 or RC-v2 operate in any way that is similar to RC-Current with respect to this functionality.

32. Instead, in RC-v1, `TorrentLoader.java`[52] searches the "C:\Users\payart2\Downloads\" directory for files ending with ".torrent". Then information is parsed from each .torrent file; this information includes the .torrent info field hash, the .torrent filename, and the number of pieces that comprise the

[52] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorentapi-v1.0/src/com/boswell/torrent/TorrentLoader.java

.torrent payload. This information and the contents of the .torrent file itself are inserted into the *torents* database table for each .torrent file. Given that the .torrent files are read from the "C:\Users\payart2\Downloads\" directory and I have seen no source code in RC-v1 that indicates that files are written to "C:\Users\payart2\Downloads" directory, it appears from the source code that RC-v1 relies on a manual process where an individual downloads .torrent files suspected to contain copyrighted works in their payloads.

33. I have seen no source code in RC-v2 that relates to the ingestion of copyrighted works, finding .torrent files on the Internet, or filtering .torrent files based on matching artist and title against filename whatsoever. In particular, there is no `TorrentLoader.java` file in RC-v2. There are also no references in RC-v1 or RC-v2 to the *tracks* database table, which is used in the ingestion process of RC-Current. All of this suggests that in both RC-v1 and RC-v2 the process of ingesting information about copyrighted works and selecting .torrent files to monitor was performed manually. Frederiksen-Cross at ¶63 confirms my understanding that this was a manual process in 2013, in the final and penultimate rows of the table included at the end of that paragraph. Boswell has also testified that in 2013, Rightscorp used manual methods for the .torrent file ingestion process.[53]

[53] Deposition of Greg Boswell, July 29, 2015, 470:14 – 18

34. I have seen no evidence in RC-v1 or RC-v2 that any sort of verification was performed on the manually downloaded .torrent files. I am also not aware of any produced documentation that suggests that there ever existed written procedures or tutorials for how the manual ingestion process should have been performed in order to maintain accuracy and consistency over time. Steele has testified that he is not aware of any tutorial materials that existed to memorialize this manual procedure.[54] In the context of processing structured data like information about copyrighted works and .torrent files, manual processes tend to be more error-prone than automated computer processes, especially because people tend to fatigue quickly when doing such repetitive tasks, whereas computers performing the same tasks in an automated way suffer from no similar shortcoming.[55] In the context of processing structured data like information about copyrighted works and .torrent files, automated computer processing also generally operates more quickly than manual processing. For these reasons it is unsurprising that Rightscorp later sought to employ an automated solution to the ingestion and .torrent file selection process. For these reasons, in comparison to RC-Current, the ingestion and .torrent file selection process of RC-v1and RC-v2 casts further doubt

[54] Deposition of Robert Steele, July 29, 2015, 33:14 – 34:12

[55] See, e.g. http://users.ece.cmu.edu/~koopman/des_s99/human/ : "Automated systems are extremely good at repetitive tasks.", "Humans are much better than machines at handling novel occurrences, but cannot perform repetitive tasks well."

on the reliability of Rightscorp's software for detection of instances of alleged infringement.

2. Verification of the Artist and Title for Music Files in the Payloads of Monitored .torrent Files

35. The operation of RC-Current with respect to the verification of the artist and title for music files in the payloads of monitored .torrent files is discussed in Rucinski ¶¶19 – 23. In particular, RC-Current utilizes some combination of the Audible Magic and AcoustID fingerprinting technologies to attempt to perform this verification. I have seen no source code that suggests that either RC-v1 or RC-v2 performs fingerprinting of music files in any way or otherwise has any automated process for performing this verification. In particular, none of the files discussed in Rucinski ¶¶19 – 23 appear in RC-v1 or RC-v2, namely `SampleIt3.java`, `SampleRequest.jsp`, `SFT.jsp`, `MusicDownload.java`, or `MusicDownloadPDCleanUp1.java`. Frederiksen-Cross at ¶63 confirms my understanding that the verification of downloaded files was a manual process in 2013, in the fourth row of the table included at the end of that paragraph, which contains "SampleIt3.java" in the second column. Greg Boswell has also testified that in 2013, Rightscorp used manual methods for the .torrent payload verification process.[56]

[56] Deposition of Greg Boswell, July 29, 2015, 468:12 – 16

36. I have seen no evidence in the produced source code that any automated sort of verification was performed on the manually verified .torrent files. I am also not aware of any produced documentation that suggests that there ever existed written procedures or tutorials for how the manual .torrent verification process should have been performed in order to maintain accuracy and consistency over time. Robert Steele has testified that he is not aware of any tutorial materials that existed to memorialize this manual procedure.[57] Robert Steele also testified that the reason for subsequently employing an automated system to perform this process was because, "there became a point where we had more copyrights to monitor than could be entered into the system manually". For these reasons, in comparison to RC-Current, the .torrent payload verification process of RC-v1and RC-v2 casts further doubt on the reliability of Rightscorp's software for detection of instances of alleged infringement during the relevant timeframe.

3. Monitoring Bitfields of IP Addresses

37. The operation of RC-Current with respect to monitoring the bitfields of IP addresses associated with peers in a BitTorrent swarm is discussed in Rucinski

[57] Deposition of Robert Steele, July 29, 2015, 30:17 – 32:13

¶¶24 – 28.[58] Both RC-v1 and RC-v2 monitor these bitfields in substantially the same way as RC-Current, with a few differences. In particular, `Test5.java` in both RC-v1 and RC-v2 calculates the "fullfile" field of the *TorrentInfractions* database table in the same way as RC-Current, where the "fullfile" field is set to 1 if every bit in the bitfield for a peer is set to 1, and the "fullfile" field is set to 0 otherwise. This compatibility with RC-Current means that `FullFilesBeing.txt` (see Rucinski ¶26) would be able to correctly process and update entries created by the `Test5.java` program in both RC-v1 and RC-v2. `Test5.java` in both RC-v1 and RC-v2 also stores the bitfield corresponding to each peer in the same way as `Test5.java` in RC-Current.[59] Despite being stored for each peer, Boswell testified that there was no reason to save the bitfield[60] since "the full load file [flag] already gets set before [for] this

[58] For clarity, I will also note that the bitfield for a given peer is stored according to line 244 of Test5.java in RC-Current. That line results in the toString method being called on a HashMap that stores information about the status of each piece. The toString method returns a very verbose representation of the bitfield that might more naturally be expressed as a simple sequence of "0"s and "1"s.

[59] Additionaly, the date "1/16/12" and time "1:04 PM" are found in a comment at the top of the `Test5.java` program in RC-v1, RC-v2, and RC-Current.

[60] It is my understanding that in the context of Greg Boswell's testimony, "bitfield" and "piece map" are synonymous. See Peer.java, ll. 307, 310, 353, 354 and SampleIt3.java, l. 244.

purpose."[61] For completeness, below I separately describe the minor differences from RC-Current for RC-v1 and RC-v2.

i. I have seen no source code in RC-v1 that relates to launching an instance of the `Test5.java` program,[62] though `InfringementFinder.sh` in RC-JLI (discussed below) launches an instance of the `Test5.java` program. It is possible that the `Test5.java` program in RC-v1 was launched manually either directly or via `InfringementFinder.sh` in RC-JLI. Frederiksen-Cross ¶6 states that to the extent there are differences between RC-v1 and RC-Current, "those differences, or changes were made to make the code run more efficiently, to automate processes..." The `Test5.java` program in RC-v1 also has a different mechanism for getting a peer's ISP in comparison to the RC-Current version of `Test5.java`.

ii. In RC-v2, the `Deamon.java` program creates a number of threads indicated by the first parameter passed to the program, and each such thread launches an instance of the `Test5.java`

[61] Deposition of Greg Boswell, July 29, 2015, 440:13 – 441:17

[62] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/Test5.java

program.[63] However, I have seen no source code in RC-v1 or RC-v2 that launches `Deamon.java`, so I cannot determine how many instances of `Test5.java` would be launched by `Deamon.java`. The `Test5.java` program in RC-v2 also has a different mechanism for getting a peer's associated ISP in comparison to the RC-Current version of `Test5.java`; the method used is RC-v2 is the same method used in RC-v1.

4. Expanding Alleged Infringements Associated with .torrent Files to Alleged Infringements Associated with Individual Songs

38. I have seen no evidence in RC-Historical of expanding alleged infringements with .torrent files to alleged infringements associated with individual songs as described in Rucinski ¶¶29 – 30 for RC-Current. Outside of RC-Historical, in RC-JLI there is some evidence of expanding alleged infringements in the `DownloadRequest.jsp`[64] program. This program queries the *ExpandedTI* database table, which is the name of a database table used in RC-Current and discussed in Rucinski ¶¶29 – 30. The presence of `DownloadRequest.jsp` in RC-JLI suggests either that RC-JLI is not contemporaneous with RC-Historical or,

[63] Rightscorp 07-15-15/RC20130909/jBittorrentAPI-1/com/boswell/torrent/Deamon.java

[64] Rightscorp 07-15-15/jli transmittal 7-2-131/jli transmittal 7-2-13/DownloadRequest.jsp

more likely, that the infringement expansion code for RC-Historical was not produced even to Barbara Frederiksen-Cross. I have seen no evidence in RC-Historical of the *torrentcopyrights* database table being read from or written to in RC-Historical, and this table is central to the infringement expansion process in RC-Current.

5. Generating Notifications of Claimed Infringement

39. I have seen no source code in RC-v1 or RC-v2 related to generating notifications of claimed infringement as described in Rucinski ¶¶31-33 for RC-Current. In particular, Greg Boswell testified that in 2013, there was a query that was built to determine repeat offenders;[65] this query has not been produced.

6. ISP Dashboard

40. I have seen no source code in RC-v1 or RC-v2 that relates to an ISP Dashboard as described in Rucinski ¶¶34-35 for RC-Current.

7. Downloading and Storing Torrent Payload Files

41. Both RC-v1 and RC-v2 download and store files from torrent payloads with the `SampleIt2.java` program in substantially the same way as each other and as RC-Current as described in Rucinski ¶¶36-37. When comparing the RC-v1 and RC-Current versions of `SampleIt2.java` during his deposition

[65] Deposition of Greg Boswell, July 29, 2015, 469:16 – 21

on July 29, 2015, Greg Boswell identified a few differences between the files. Some of them relate to insignificant differences between the files, but Greg Boswell also testified that the RC-v1 version of `SampleIt2.java` was less aggressive in making sure that all the pieces of a payload file had been downloaded from the given peer.[66] Greg Boswell also testified that the RC-v1 version of `SampleIt2.java` lacked the verification that it had gotten all the pieces from the peer at the end of its process.[67] Because the RC-v1 version of `SampleIt2.java` lacks these verification steps, its operation introduces further concerns about the reliability of Rightscorp's software for detecting instances of alleged infringement during the relevant time period.

C. Fragmentary Productions Within RC-JLI

42. The RC-JLI production mainly contains JSP web server programs that serve as interfaces between the Rightscorp system's Java programs and the database tables storing the data collected by those Java programs. The only file that is not a web server program is `InfringementFinder.sh`, which is discussed below. It is unclear why these files were organized separately from RC-v1 an RC-v2 since they are necessary, but not sufficient, for the complete functioning of the Rightscorp system.

[66] Deposition of Greg Boswell, July 29, 2015, 431:18 – 431:22

[67] Deposition of Greg Boswell, July 29, 2015, 431:23 – 432:3

i. The `Detected.jsp`[68] program in RC-JL has no functional differences in comparison to the version of this file in RC-Current as described in Rucinski ¶¶25-26. The differences between the RC-JLI and RC-Current versions of the `Detected.jsp` program are that more debugging and status information was printed to the output terminal in the RC-JLI version, and many comments have been removed in the RC-Current version.

ii. The `DownloadRequest.jsp` program in RC-Current, as described in Rucinski ¶36, selects a random row from the *GetThisInfringer* database table joined with the *torents* database table and sends the IP address, port number, file name, and .torrent file back to the `SampleIt2.java` program. The `DownloadRequest.jsp`[69] program in RC-JLI ultimately sends the same information to the `SampleIt2.java` program, but the query it uses to get that information is different. The `DownloadRequest.jsp` program in RC-JLI selects a random

[68] Rightscorp 07-15-15/jli transmittal 7-2-131/jli transmittal 7-2-13/Detected.jsp

[69] Rightscorp 07-15-15/jli transmittal 7-2-131/jli transmittal 7-2-13/DownloadRequest.jsp

row from the *ExpandedTI* database table joined with the *torents* database table based on their torrent hashes being equal.

iii. The `InfringementFinder.sh`[70] program in RC-JLI launches the `Test5.java` program with no parameters.

iv. The `IPRequest.jsp`[71] program in RC-JLI responds to requests which contain an IP address parameter. The `IPRequest.jsp` program queries the *EmailRegestry* database table for the number of rows that match the IP address parameter, have a port that is not "null", and have a "theday" field equal to the current day. If there are no rows matching these criteria, then the `IPRequest.jsp` program responds to the request with "true" and otherwise responds with "false".

v. The `TFT.jsp`[72] program in RC-JLI performs substantially the same function as the RC-Current version of TFT.jsp, as described in Rucinski ¶36.

vi. The `TorrentRequest.jsp`[73] program in RC-JLI functions in a somewhat similar manner to the `TorrentRequest.jsp`

[70] Rightscorp 07-15-15/jli transmittal 7-2-131/jli transmittal 7-2-13/InfringementFinder.sh

[71] Rightscorp 07-15-15/jli transmittal 7-2-131/jli transmittal 7-2-13/IPRequest.jsp

[72] Rightscorp 97-15-15/jli transmittal 7-2-131/jli transmittal 7-2-13/TFT.jsp

program in RC-Current, which is discussed in Rucinski ¶¶24, 28. The difference is that the `TorrentRequest.jsp` program in RC-JLI selects a random row from the *torents* database table, as opposed to the *torrentcopyrights* database table in the RC-Current version. In addition to that difference, by using a random number generator, the `TorrentRequest.jsp` program in RC-JLI will override the random row selection five percent of the time and select the row with a hash matching a specific hash that is literally given in the source code at line 32.

D. **Superfluous Productions Within RC-v1**

43. There are a number of files in RC-v1 that I have decided to treat as miscellaneous files because they do not appear in RC-Current. These files are: `DownloadIt.java`, `DPTest.java`, `InfringementKeys.java`, `IPProcessor2.java`, `MT.java`, `parcetest.java`, `ReportingInterface.java`, `SampleIt.java`, `SocketSystem.java`, `systest.java`, `test.java`, `test2.java`, `Test3.java`. `TorrentLoader.java`, `Traffic.java`, and `UDPFun.java`. These files are

[73] Rightscorp 07-15-15/jli transmittal 7-2-131/jli transmittal/7-2-13/TorrentRequest.jsp

either non-production test code, perhaps predecessors to programs that persisted in RC-Current, or perhaps files that predate Boswell's involvement with Rightscorp.

44. In addition to the `Test5.java` program produced in RC-v1, there are three other files in RC-v1 that appear to be previous versions of the RC-v1 `Test5.java` program. These files are `test.java`, `test2.java`, and `Test3.java`; I discuss each of these files below.

i. The `test.java`[74] program gets a list of files ending in "test.torr" from the "<user.home>\Downloads\" directory, where "<user.home>" is the home directory of the user currently logged into the computer.[75] `test.java` then parses each file in the list of files into a jBittorrentAPI TorrentFile object. For each TorrentFile object, `test.java` then uses the jBittorrentAPI library to collect a list of peers from the tracker associated with each .torrent file over a period of six minutes.[76] `test.java` inserts information about each peer, including the peer's IP address and port number, the bitfield sent by each peer, the date and time at

[74] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorentapi-v1.0/src/com/boswell/torrent/test.java

[75] This is determined by calling the System.getProperty method with a parameter of "user.home". See lines 39 and 49 of `test.java`.

[76] See `test.java` at line 53.

which each peer was contacted, and a randomly generated UUID[77] into the *TorrentInfractions* database table. Unlike the `Test5.java` program in RC-v1, which sends this information to the `Detected.jsp` program to be inserted into the database, `test.java` inserts the information into the database directly. The `test.java` program deals with the "fullfile" field and the bitfield sent by the peer in the same way as `Test5.java` in RC-v1, as described above. Given that the `test.java` program deals only with torrents with filenames ending in "test.torr", it seems likely that the `test.java` program is only a test program and was not part of the production code in RC-v1. Boswell has also testified that the `test.java` program was never in production in Rightscorp's system.[78]

[77] https://en.wikipedia.org/wiki/Universally_unique_identifier; the term Gloabally Unique Identifier ("GUID") refers to certain implementations of the general UUID standard. For the purposes of this report and Rightscorp's system, there is no practical difference between UUID and GUID.

[78] Deposition of Greg Boswell, July 29, 2015, 434:23 – 435:11

ii. The `test2.java`[79] program uses the jBittorrentAPI library to get a list of peers from the tracker associated with a specific .torrent file.[80] For each peer in the peer list: `test2.java` gets the peer's ID, the peer's IP address, and the peer's port. A socket is then opened to the peer's IP address and port. The `test2.java` program then uses the jBittorrentAPI library to set up a message sender and a message receiver on the connection to the peer. An OutgoingListener is added[81] to the message receiver in order to handle the various types of BitTorrent protocol messages. In response to most BitTorrent protocol messages, the OutgoingListener only prints a string representing information about the state. The only states for which the OutgoingListener does more than that is for the BitTorrent protocol messages "HAVE", "BITFIELD", and "PIECE". In response to the "BITFIELD" message, the `test2.java` program saves the bitfield indicating which payload pieces the peer is in possession of

[79] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorentapi-v1.0/src/com/boswell/torrent/test2.java

[80] See `test2.java` at line 202.

[81] See `test2.java` at line 61.

at the time of the message. In response to the "HAVE" message, test2.java updates the bitfield for the peer to indicate that the peer now has a specific piece. In response to the "PIECE" message, the `test2.java` program saves part of the data from the received message to the "data" variable but does not do anything further with the "data" variable. The `test2.java` program does not record the information it encounters into any database tables, indicating that it is most likely a test program and was not part of the production code in RC-v1. Boswell has testified that the `test2.java` program was never in production and was used instead for his own education.[82]

iii. The `Test3.java`[83] program reads a hard-coded .torrent file from "C:\Users\payart2\Downloads\[isoHunt] Katy Perry – Teenage Dream [2010 – MP3 – 320 kbps] [vigoni].torrent" on the host computer. The `Test3.java` program prints information about the .torrent file and then uses the jBittorrentAPI library to collect a list of peers from the tracker associated with the .torrent file. The

[82] Deposition of Greg Boswell, July 29, 2015, 436:2 – 436:11

[83] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorentapi-v1.0/src/com/boswell/torrent/Test3.java

`Test3.java` program does not record any information it gathers into any database tables, indicating that it is most likely a test program and was not part of the production code in RC-v1. Boswell has testified that the `Test3.java` program was never used in production.[84]

45. The `SampleIt.java`[85] program in RC-v1 includes some of the functionality of the SampleIt2.java program, including connecting to a peer and downloading pieces from a peer. Boswell has testified that the SampleIt.java program was never used in production and that its purpose was educational.[86]

46. The `InfringementKeys.java` program enables the parsing of a response from the American Registry for Internet Numbers ("ARIN").[87]

47. The `IPProcessor2.java` program makes requests to ARIN for the information associated with a given IP address, e.g. the owner and whom to contact regarding the IP address, and then uses the `InfringementKeys.java`

[84] Deposition of Greg Boswell, July 29, 2015, 438:20 – 439:1

[85] Rigthscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/SampleIt.java

[86] Deposition of Greg Boswell, July 29, 2015, 433:5 – 19

[87] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/InfringementKeys.java; Deposition of Greg Boswell, July 29, 2015, 413:21 – 414:1

program to parse the response.[88] Greg Boswell testified that this file was probably developed prior to his work at Rightscorp and that its functionality was not specific to BitTorrent.[89]

48. The `ReportingInterface.java` file defines an interface with a "reportBack" method to be implemented by other Java programs. Greg Boswell testified that the `ReportingInterface.java` file was primarily used during the debugging and development phases of the Rightscorp system.[90]

49. The following files in RC-v1 performed utility or test functions.

i. The `DownloadIt.java`[91] program gets all the files that end in ".torrent" in the "<user.home>\Downloads\" directory, where "<user.home>" is the home directory of the user currently logged into the computer.[92] The `DownloadIt.java` program then uses the jBittorrentAPI library to parse and print information about each

[88] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/IPProcessor2.java; Deposition of Greg Boswell, July 29, 2015, 416:7 – 416:16

[89] Deposition of Greg Boswell, July 29, 2015, 416:8 – 19, 418:6 – 12

[90] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/ReportingInterface.java; Deposition of Greg Boswell, July 29, 2015, 423:16 – 424:18

[91] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/DownloadIt.java

[92] This is determined by calling the System.getProperty method with a parameter of "user.home". See lines 34 and 43 of `DownloadIt.java`.

.torrent file and collect a list of peers from the tracker associated with each .torrent file. `DownloadIt.java` does nothing further with the information it collects. Boswell testified that this program probably ran at one point in time, around 2011.[93]

ii. The `DPTest.java`[94] program performs a very short datagram packet test; however there is no evidence of it being instantiated or run. Boswell testified that `DPTest.java` was never in production and that it was a diagnostic file to test the BitTorrent protocol.[95]

iii. The `MT.java`[96] program implements an SSL trust manager as defined by javax.net.ssl.X509TrustManager, but the method implementations do nothing beyond simply printing out parameters or returning default values. Boswell testified that this file was not specifically developed for Rightscorp but that it was probably used at one point in time during development testing.[97]

---

[93] Deposition of Greg Boswell, July 29, 2015, 399:24 – 400:18

[94] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/DPTest.java

[95] Deposition of Greg Boswell, July 29, 2015, 408:20 – 409:22

[96] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/MT.java

[97] Deposition of Greg Boswell, July 29, 2015, 419:1 – 420:5

iv. The `parcetest.java`[98] program simply tests a regular expression to see if it can parse the string "UdP://tracker.torrentbay:to:6969/announce". Boswell testified that parcetest.java was an educational file developed in order to understand the shift of BitTorrent trackers from HTTP to UDP.[99]

v. The `SocketSystem.java`[100] program defines a method which returns a socket to a given URL.

vi. The `systest.java`[101] program prints system environment variables and properties.

vii. The `Traffic.java` program launches the Wireshark[102] program to monitor traffic on a UDP port of the host machine, but I have not seen any evidence in the produced source code that the `Traffic.java` program is actually executed.[103]

---

[98] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/parcetest.java

[99] Deposition of Greg Boswell, July 29, 2015, 421:16 – 422:6

[100] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/SocketSystem.java

[101] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/systest.java

[102] https://www.wireshark.org/

[103] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/Traffic.java

viii. The `UDPFun.java` program is a collection of UDP and string parsing test code that Greg Boswell used to test the transition of Rightscorp's system from HTTP BitTorrent trackers to UDF BitTorrent trackers in the beginning or middle of 2013.[104]

V. RIGHTSCORP'S SOURCE CODE AND DATA PRODUCTION REMAINS INADEQUATE

50. Rightscorp's source code production has generally continued to be inadequate. Due to this continued inadequacy, I have not been able to perform timely analysis of much of the code and data that was eventually produced, and I have not been able to analyze source code and the data that affected its operation during all relevant times. The vast majority of the relevant time period has no corresponding source code or data produced to demonstrate the functionality of the Rightscorp system over that time period. The few historic source code files Rightscorp has produced do not even form one fully functional version of the Rightsorp system. In this section I enumerate the various ways that Rightscorp has failed to produce all relevant requested source code and data in a reasonably timely manner and in a reasonable format.

51. Rightscorp has failed to produce source code and data for files and databases that change frequently in ways that materially affect their operation. For

[104] Rightscorp 07-15-15/rightscorp/rightscorp/jbittorrentapi-v1.0/src/com/boswell/torrent/UDPFun.java; Deposition of Greg Boswell, July 29, 2015, 353:2 – 354:9

example, as discussed above, Boswell modified the contents of the stored procedure shown in `FullFilesBeing.txt` so that, as produced, it would not be automatically executed by a database event. As discussed in above, Boswell also modified the contents of the `CoxMusicDownloader.java` program so that, as produced, it would not accurately store the music samples that were produced on the Hard Drives. As another example discussed in above, Boswell modified the contents of the *AutoTermContactList* database table, which materially affects the operation of the `SampleIt2.java` program; to my knowledge no versions of the data in the *AutoTermContactList* database table, historical or otherwise, have been produced. The amount of manual, ad hoc changes that appear to happen on a fairly regular basis in the operation and maintenance of the Rightscorp system make it difficult to analyze, but without a record of the changes, analyzing how the system operated at any point in the past is impossible.

52. Rightscorp has very belatedly produced some historic versions of its source code in the form of RC-v1, RC-v2, and RC-JLI, but those versions are even more incomplete than the production set for RC-Current. I have summarized what has and has not been produced regarding the historical versions RC-v1, RC-v2, and RC-JLI above. None of these directories of historic code can stand alone as a version of the Rightscorp system. It may be possible that a particular program in RC-v1 or RC-v2, can be compiled, but it would not function without its supporting

programs in RC-JLI. Even considering all of RC-v1, RC-v2, and RC-JLI, no functioning historical version of the Rightscorp system could be assembled. The only way to get information about historic files that were not produced is from the memories of those who might have seen them, but that is not a reliable record. [105]

53. Rightscorp has also only recently produced various source code files that were obviously relevant to previous requests for source code production since the source code files recently produced were clearly related to source code files produced earlier. For example, I received the `CEmail.java` program on May 26, 2015. As discussed above, the `CPost.html` web page is the only way that the `CEmail.java` program is executed; I only received the `CPost.html` program on July 15, 2015. Along with the `CEmail.java` program, I received on May 26, 2015 a template[106] for emails that are sent out by the `CEmail.java`. Part of this email template is a settlement offer that gives the recipient of the email a URL to a web page through which the recipient can pay Rightscorp (see the discussion above). However, I did not receive the source code for this web page, `Payment.java`, until July 15, 2015.

54. Finally, in response to the identification of various source code production deficiencies pointed out in Rucinski, Frederiksen-Cross states in a

[105] Deposition of Greg Boswell, July 29, 2015, 469:16 – 24

[106] 2015.05.26 – Additional Rightscorp Source Code/CoxEmailCode/cox.txt

number of places that certain source code files have been produced. Frederiksen-Cross states at ¶22, "It is my understanding that RightsCorp has produced additional source code, specifically including the source for portions of the Rightscorp system that was made available to me in 2013." Frederiksen-Cross states at ¶24, "[I]t is my understanding that the RightsCorp code I examined in 2013 has been produced." Frederiksen-Cross states at ¶27, "In paragraph 46 Rucinski notes that he has not seen any code that invokes CEmail. This code has been produced." For clarity, in each of these instances, the referenced source code was produced after I submitted Rucinski, and I therefore did not have access to any of the source code referenced in these instances while performing analysis for that report.

VI. THE TORRENT PAYLOAD SAMPLES PRODUCED BY RIGHTSCORP DURING GREG BOSWELL'S DEPOSITION ON JULY 3, 2015, ARE NOT A RELIABLE EVIDENTIARY RECORD OF MUSIC THAT BITTORRENT PEERS ALLEGEDLY OFFERED.

A. Generation and Production of the Torrent Payload Samples

55. During the deposition of Greg Boswell on July 3, 2015, two hard drives (collectively, the "Hard Drives") were produced to Cox's counsel. Boswell testified that he used the "Cox music downloader" software that he developed in order to download the samples produced on the Hard Drives.[107] The *TorrentSampleIndex* and *TorrentSamples* database tables are populated directly by

[107] Depositon of Greg Boswell, July 3, 2015, 104:9 – 105:13

`TFT.jsp`, which is utilized by `SampleIt2.java` to store sampled files and metadata related to those sampled files (see Rucinski ¶36). While the *TorrentSamples* database table stores both files from torrent payloads as well as metadata about those files, the *TorrentSampleIndex* database table stores only metadata about files from torrent payloads.[108]

56. `CoxMusicDownloader.java`[109] connects to the *TorrentSampleIndex* and *TorrentSamples* database tables. However, as produced, the `CoxMusicDownloader.java` program could not have generated the samples on the Hard Drives; according to lines 21 and 23, the greatest number of samples that could have been generated by it is 479,500, and yet more than 702,000 samples were produced on the Hard Drives. Boswell testified that he modified the code in the course of producing the samples,[110] so the produced code reflects only the most recent version of the `CoxMusicDownloader.java` program. Boswell also testified that the operation of the `SampleIt2.java` program relies on the manual configuration of the *AutoTermContactList* database table in order to target a particular ISP.[111] To my knowledge no current or historic versions of the *AutoTermContactList* database table have been produced even though the contents

[108] 20150609/TFT.jsp, RIGHT_SC00000130–131

[109] Rightscorp20150627/20150627/CoxMusicDownloader.java

[110] Deposition of Greg Boswell, July 29, 2015, 388:9 – 18, 390:4 – 13

[111] Deposition of Greg Boswell, July 29, 2015, 397:3 – 398:22

of this table directly affects what data the `SampleIt2.java` program operates on and therefore the data that the `CoxMusicDownloader.java` program considers when collecting music samples.

B. Analysis of Samples on the Hard Drives

57. I have analyzed the files on the Hard Drives (see Exhibit C) provided by Rightscorp along with the data from the *infractions* database table, the data from the *TorrentSampleIndex* database table, and the list of asserted works.[112] A summary of my analysis is in Exhibit B. Exhibit B lists the number of records in the *infractions* database table and the number of files on the Hard Drives. The source code and CSV data I used to perform this analysis is included with this report as RUCINSKI_00000028.

1. Aggregate Data

58. The first column in the table lists the number of records in the *infractions* database table. Of the 14,817,130 records, 10,621,115 are infraction records related to Rightscorp's detection of alleged infringements before November 26, 2014 (the date on which Plaintiffs filed their original complaint), and 4,196,015 are infraction records relating to Rightscorp's detection of alleged infringements on or after November 26, 2014.

[112] See the Schedule Bs appended to Plaintiff's respective First Supplemental Responses to Defendant CoxCom, LLC's Interrogatory Nos. 1, 2, 8, and 9.

59. The second column in the table lists the number of records in the *infractions* database table that appear to correspond to musical compositions I understand Plaintiffs assert in this lawsuit.[113] Of the 2,544,846 records, 2,022,313 are infraction records related to Rightscorp's detection of alleged infringements before November 26, 2014, and 522,533 are infraction records related to Rightscorp's detection of alleged infringements on or after November 26, 2014.

60. The third column in the table lists the number of files on the Hard Drives. There are 702,201 files on the Hard Drives that appear to have been downloaded from peers by Rightscorp's system (specifically utilizing the `SampleIt2.java` program and the `CoxMusicDownloader.java` program). These files have 657,234 unique infractionguid values. 499,024 of these files have an infractionguid that is in a row in the *infractions* database table, while 203,177 of these files have an infractionguid that is not in any row in the *infractions* database table.

[113] To narrow the records in the *infractions* database table for the musical compositions at issue in this case, I filtered the *infractions* database table rows so that the artist and title fields in that table match the artist and title for the asserted musical compositions. The spreadsheet containing the information about the asserted musical compositions appears as RUCINSKI-00000028; the information appearing in this table was derived from the Schedule Bs appended to Plaintiff's respective First Supplemental Responses to Defendant CoxCom, LLC's Interrogatory Nos. 1, 2, 8, and 9.

61. The fourth column in the table lists the number of files on the Hard Drives where the artist and title from the corresponding row in the *infractions* database table (based on matching infractionguid values) that appear to correspond to the list of musical compositions I understand Plaintiffs assert in this lawsuit.[114] As a consequence of the matching of the files on the Hard Drives to the list of works using the title and artist from the *infractions* database table, all of these files have an infractionguid that is in a row in the *infractions* database table.

62. Given the above data, in particular the surprisingly low percentage of samples that match asserted musical compositions, the torrent payload samples produced by Rightscorp during Greg Boswell's deposition on July 3, 2015, are not a reliable evidentiary record of music that BitTorrent peers allegedly offered.

2. Specific Anomalous Examples

63. There are numerous instances where the song title in the filename of a file on the Hard Drives does not match the title in the *infractions* database table for the infractionguid of the file on the Hard Drives. For example, there are 108 files on the Hard Drives with the filename "Aerosmith – The Essential Aerosmith CD2 (2011)02 – Livin' On the Edge.mp3" and with a corresponding row in the *infractions* database table (matching on infractionguid). All 108 corresponding rows

[114] I used a similar methodology for finding rows that corresponded to the list of musical works that I used for column 2.

in the *infractions* database table have "Aerosmith" as the artist. However, 61 of the 108 rows have "Mama Kin" as the title. 47 of the rows have "Same Old Song and Dance" as the title. None of the rows have any variation of "Livin' On the Edge" as the title.

64. In addition to the music files on the Hard Drives there are several files that do not appear to contain music. There are 122 files that appear to be images (14 PNG files and 108 JPEG files, with an extension of ".jpg", ".JPG", or ".jpeg"),[115] six PDF files, and 559 files that appear to be video files (with extensions ".avi", ".m4v", ".mkv", ".mp4", ".MP4", ".mpg", and ".VOB").

65. For example, the six PDF files appear to be six copies of the same PDF file, all downloaded on May 2, 2014 (based on the "putin" field of the corresponding row in the *TorrentSampleIndex* database table). Three of these copies have an infractionguid of "TC-5e3a3842-ebec-4ba9-aefc-f3d052566cda". These three copies of the PDF files are the only files on the Hard Drives with this infractionguid. This infractionguid is found in one row in the *infractions* table, which has "017 – Foo Fighters – Let It Die [Torrent Tatty] (? Roswell RCA).MP3" for the filename, "Foo Fighters" as the artist, and "Let It Die" as the title.

[115] There are also an additional 36 files on the two hard drives, each of which are named either "AlbumArtSmall.jpg" or "Folder.jpg". These files were most likely automatically created by the Windows operating system when the hard drives were connected to a computer; they are most likely not files downloaded from peers by Rightscorp's system.

Two of the other copies of the PDF file have an infractionguid of "TC-3dcd8824-ad75-4116-8255-8a3dc50ff461", with the corresponding row in the *infractions* database table having "039 – MGMT – Kids [Torrent Tatty] (? Columbia).MP3" for the filename, "MGMT" as the artist, and "Kids" as the title. These two copies of the PDF files are the only files on the Hard Drives with this infractionguid. The final copy of the PDF file has an infractionguid of "TC-f0e0716a-c8da-4b7e-beab-a6e51ebd986c", with the corresponding row in the *infractions* database table having "002 – Kings of Leon – Sex On Fire [Torrent Tatty] (? RCA).MP3" for the filename, "Kings of Leon" as the artist, and "Sex On Fire" as the title. This copy of the PDF file is the only file on the Hard Drives with this infractionguid.

66. Given the above anomalies, the torrent payload samples produced by Rightscorp during Greg Boswell's deposition on July 3, 2015, are not a reliable evidentiary record of music that BitTorrent peers allegedly offered.

VII. RESPONSE TO REPLY OF BARBARA FREDERIKSEN-CROSS

67. Frederiksen-Cross is inconsistent on the point of whether the `SampleIt3.java` program downloads files from a single peer or multiple peers, though Frederiksen-Cross more frequently states that the `SampleIt3.java` program downloads files from a single peer. In five separate paragraphs,[116]

[116] See Frederiksen-Cross at ¶¶10, 28, 31, 32, and 45

Frederiksen-Cross states that the `SampleIt3.java` program[117] downloads a file from a single peer, and in ¶28 Frederiksen-Cross cites specific analysis in support of this assertion. I disagree with that analysis, and I provide more detail below. My analysis is consistent with ¶34 of Frederiksen-Cross and ¶60 of the Report of Barbara Frederiksen-Cross, submitted on June 19, 2015, which both state that the `SampleIt3.java` program downloads files from "randomly selected peers." Greg Boswell has now testified twice that the `SampleIt3.java` program downloads from the swarm associated with a .torrent file and does not download from a single peer in the swarm;[118] This is consistent with my analysis as well.

i. Frederiksen-Cross at ¶28 states specifically, "The SampleIt3.java processing that downloads all pieces of a file from a single peer is accomplished via the override of the TrackManager class that begins at line 516 of SampleIt3. The download of file pieces from a single connected peer occurs in the extension to method PeerReady which starts at line 526 of SampleIt3." I agree that the cited TrackManager class is involved in the download of pieces of a .torrent file that correspond to a single file in a .torrent file. I

[117] Frederiksen-Cross at ¶10 does not explicitly mention `SampleIt3.java` but does reference Rucinski at ¶19, which discusses `SampleIt3.java`.

[118] Deposition of Greg Boswell, July 3, 2015, 165:15 – 166:22, 186:13 – 187:3, 247:23 – 249:5; Deposition of Greg Boswell, July 29, 2015, 426:21-24

disagree that the peerReady method defined at line 526 of `SampleIt3.java` restricts the download of pieces of a .torrent file to downloading from a single peer. Below I explain the operation of the peerReady method in the context of the `SampleIt3.java` program and the jBittorrent code that Rightscorp has produced. Rucinski at ¶19 provides an overview of the broader process.

ii. The peerReady method of the TrackManager class overrides the peerReady method of its superclass, DownloadManager. The peerReady method of the DownloadManager class is defined at line 659 of `DownloadManager.java`. It utilizes the choosePiece2Download method of the DownloadManager class to pick a random piece from the set of remaining pieces in the torrent for the given peer to download. Since the overall purpose of `SampleIt3.java` is to download a particular file rather than all of a torrent, it makes sense that Rightscorp overrode this method with the peerReady implementation in the TrackManager class.

iii. The peerReady method of the TrackManager class is defined at line 526 of `SampleIt3.java`, and it takes as its only parameter an identifier for the peer for which a piece to download needs to be

selected. The peerReady method of the TrackManager class is called each time a peer in the swarm is ready to be downloaded from, and it is called to select a piece to download. Lines 534 through 555 populate the "block" ArrayList with the list of pieces of the current .torrent file that are currently associated with DownloadTasks[119] and are therefore already in the process of being downloaded. Lines 561 through 588 (the end of the method) iterates in order through each piece that comprises part of the file that `SampleIt3.java` is downloading. For each piece, it checks whether the piece is in the "block" ArrayList and is therefore already being downloaded. Once it finds a piece that is not in the "block" Arraylist, it calls the requestPiece method at line 569 on the DownloadTask associated with the identifier for the peer for which the piece to download needs to be selected, and then ends the method.[120] To summarize, the peerReady method selects the first piece of the file that `SampleIt3.java` is trying to

[119] Rightscorp Source Code/jBittorrentAPI/DownloadTask.java, RIGHT_SC00000238

[120] At line 572, the "i" variable is set to 500,000, which likely exceeds the number of pieces being iterated over in the loop defined at line 561. I'm not sure why a simple "break" statement wasn't used here instead, but in all likelihood the outcome is the same: the loop ceases iterating through its body after executing this line.

download that is not already being downloaded from other peers and then instructs the given peer to download that piece. The peerReady method does not select only one peer from which to download.

68. As discussed in Rucinski at ¶11, 26, 27, 32, 43, and 62, the current version of Rightscorp's software records an alleged copyright infringement and will subsequently send a notification of claimed infringement if a given IP address is willing to offer at least 10% of the torrent payload to Rightscorp. There are a number of places where Frederiksen-Cross[121] addresses discussions in Rucinski regarding the 10% bitfield threshold, where Frederiksen-Cross appears to argue that it is possible and likely that a peer that indicates a bitfield of at least 10% will later offer all of the pieces that comprise the given torrent payload. Frederiksen-Cross never quantifies exactly how likely this is argued to occur. Regardless, Frederiksen-Cross does not state that Rightscorp ever verifies that any peers ever offer the remaining pieces of the payload, and I have seen no evidence to suggest that the Rightscorp system does this. Frederiksen-Cross justifies the usage of the 10% threshold by stating that it is used to combat the lazy bitfield technique (see at least Frederiksen-Cross ¶12). Boswell testified that he was researching utilizing the HAVE message of the BitTorrent protocol, presumably to get a complete picture on

[121] See Frederiksen-Cross at ¶¶14, 15, 19, 25, and 41.

the pieces that a peer might have in addition to the insufficient 10% bitfield throshold.[122]

69. Frederiksen-Cross at ¶13 states, "[Mr. Rucinski] ignores the fact that BitTorrent, by design, is a file sharing protocol that is directed to downloading and sharing all pieces of a file or torrent payload." Frederiksen-Cross includes footnote #2 at the end of that sentence, which states, "Rucinski acknowledges this requirement in paragraph 9 of his report, where he states: 'a peer will need to obtain all of the pieces corresponding to a file in the BitTorrent payload before the user can use the file in its entirety.'" I disagree that ¶9 of Rucinski supports the quoted statement of Frederiksen-Cross at ¶13. In particular, the simple generic statement of fact that a peer (and indeed, any computer) cannot use a file in its entirety until it possesses the file in its entirety does not bear on the purpose of the BitTorrent protocol. In particular, it is certainly plausible that for certain .torrent payload comprising multiple files, peers would be interested in acquiring certain files in the torrent payload but not others.[123]

70. Frederiksen-Cross states at ¶29: "I also disagree with the characterization of SampleIt2 in paragraph 48 as not directly related to detecting

[122] Deposition of Greg Boswell, July 29, 2015, 437:15 – 438:17

[123] For example, see http://www.techsupportalert.com/utorrent-help-select-files-to-download, which includes instructions on how to select or deselect files to download when using the µtorrent BitTorrentClient: "You may deselect the files you do not want by unticking the box net to the file."

instances of alleged infringement. The SampleIt2 program is not directly involved in the process of generating Infringement Notices, but it is used to download samples of monitored works offered by the BitTorrent peers identified during the processing performed by `Test5.java`. (I.e. The peers for whom Infringement Notices were sent.)" I agree with the second sentence of ¶29. The `SampleIt2.java` program does download files from torrent payloads from peers previously identified by the `Test5.java` program, but it does not, in agreement with Frederiksen-Cross, affect the generation of notifications of claimed infringement. A given notification of claimed infringement will be sent or not sent for a particular row in the *infractions* database table independent of whether `SampleIt2.java` samples the corresponding song. In fact, from the production I have seen, only a small subset of notifications of claimed infringement ever have a corresponding music file downloaded by `SampleIt2.java` (see Exhibit D).

71. Frederiksen-Cross states at ¶61: "Based on my review of the code to date, it is my opinion that there are no significant functional differences that affect the quality of the sampling and infringement detection in the core processes (i.e., Infringement Finder and SampleIt) of the 2013 code that I reviewed, and the code already in Cox's possession." I discuss the differences between the RC-v1, RC-v2, RC-JLI, and RC-Current versions of the Test5.java, SampleIt2.java, and other programs that are necessary for the Rightscorp system to function earlier in this

report. Importantly, there are significant differences between the RC-v1, RC-v2, RC-JLI, and RC-Current versions of many parts of the Rightscorp system, which I discuss earlier in this report. In particular, the ingestion process was a manual process in 2013, and there was no fingerprinting to verify the downloaded files. I do not know the details of those manual processes, and they were an important aspect of the Rightscorp system at the time; errors in these processes would affect the overall accuracy of the system.

72. Frederiksen-Cross at ¶34 misinterprets the criticism made in Rucinski ¶¶52 and 52.i. The primary criticism is that in ¶62 of Report of Barbara Frederiksen-Cross, submitted on June 19, 2015, the reader is led to believe that the `SampleIt3.java` program takes as input the output of Infringement Finder (`Test5.java`). As explained in Rucinski ¶¶19 – 28, `Test5.java` only operates on .torrent files that have had at least one of their payload files successfully processed by `SampleIt3.java` and the subsequent fingerprinting process.

73. Frederiksen-Cross admits that certain assertions made in the Report of Barbara Frederiksen-Cross, submitted on June 19, 2015, were based solely on representations received from Rightscorp personnel and were not based on independent analysis of source code. Frederiksen-Cross states at ¶35, "In paragraph 52.ii Rucinski also states that there is no reference in SampleIt2.java or SampleIt3.java to a ratio that affects the number of files downloaded. The

information about sampling ratio was provided to me verbally by Robert Steele." Frederiksen-Cross states at ¶36 states, "In paragraph 53 of his report Rucinski states that he has found no code to limit SampleIt3's sample collection to a maximum of two songs per peer per torrent per run. This was information provided to me by Mr. Steel [sic] and was associated with what I understood to be manual processes used to control the sampling, in order to specifically to direct the sampling process with respect to which songs to search for." However, Steele has stated that he does not remember making such statements during any such conversations.[124] To my knowledge, there is no other record of these conversations, and I have found no evidence to support their substance, so I therefore maintain my disagreement with the Report of Barbara Frederiksen-Cross, submitted on June 19, 2015, as explained in Rucinski ¶¶52.ii and 53.

74. Frederiksen-Cross at ¶39 misinterprets the criticism made in Rucinski ¶57. The primary criticism is that in ¶¶54-55 of Report of Barbara Frederiksen-Cross, submitted on June 19, 2015, the `TorrentInfractionExpander.java` file is said to, "identif[y] the *.torrent files that contain works on the list of protected works." As explained in Rucinski ¶18, it is the `AutoTorrentTransferCode.java` program, which attempts to match .torrent files "that contain works on the list of protected works", i.e. in the

[124] Deposition of Robert Steele, July 29, 2015, 47:20 – 49:14

tracks database table. The `TorrentInfractionExpander.java` file does not do this, regardless of whatever else the Report of Barbara Frederiksen-Cross, submitted on June 19, 2015, states that it also does.

75. Regarding Frederiksen-Cross at ¶40, no rebuttal is made to the statements in Rucinski ¶62; the points are merely restated. I agree with the points stated in Rucinski ¶62.

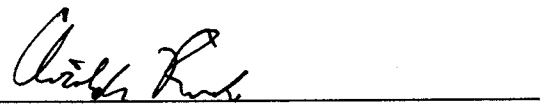
76. Frederiksen-Cross at ¶43 misinterprets my criticism in Rucinski at ¶64. My criticism was not that the language in ¶49 of Report of Barbara Frederiksen-Cross, submitted on June 19, 2015, could mislead a reader into thinking that the *MasterTorrentFiles* database table stores actual pieces from a .torrent payload. My criticism was that the reader might be misled into thinking that the *MasterTorrentFiles* database table stores rows that correspond to pieces in any way. The rows of the *MasterTorrentFiles* database table correspond to files within a .torrent payload, not pieces.

VIII. FURTHER WORK

77. I may prepare visual aids to demonstrate various aspects of my testimony at trial. I may also review additional materials produced or otherwise prepared by parties or experts in this case. I reserve the right to supplement my opinions if Plaintiffs or their experts or other parties related to this case produce new materials or source code and/or if Court decisions require me to clarify or

provide additional support for my opinions. In particular I reserve the right supplement my opinions in response to any further opinions that Ms. Frederiksen-Cross offers as outlined in Frederiksen-Cross ¶64. I also reserve the right to supplement this report based on additional documents or information, including but not limited to additional deposition testimony, additional claims asserted by Plaintiffs, or if Plaintiffs' experts offer any opinions.

Date: July 31, 2015



Christopher Rucinski

EXHIBITS A-D

INTENTIONALLY OMITTED